Darcy Ma

08/15/20

IT FDN 110: Foundations of Programming

Assignment One: Module Five

**Introduction:**

In this week, we will learn how to edit existing code. Specifically, we will learn how to replace inner data structures by dictionaries, add functionalities to load existing data, and how to add functionalities to delete entries. Then, we will verify that this script works in our IDE, Spyder. Then, we will upload to GitHub.

**Replacing Inner Data Structures by Dictionaries:**

Firstly, let us look at the code that we are going to be editing. This is a sample code provided by Professor Dirk Bisienger in the CDStarterInventory.txt file.

1. #------------------------------------------#
2. # Title: CDInventory.py
3. # Desc: Starter Script for Assignment 05
4. # Change Log: (Who, When, What)
5. # DBiesinger, 2030-Jan-01, Created File
6. #------------------------------------------#
8. # Declare variabls
10. strChoice = '' # User input
11. lstTbl = []  # list of lists to hold data
12. # TODO replace list of lists with list of dicts
13. lstRow = []  # list of data row
14. strFileName = 'CDInventory.txt'  # data storage file
15. objFile = None  # file object
17. # Get user Input
18. **print**('The Magic CD Inventory\n')
19. **while** True:
20. # 1. Display menu allowing the user to choose:
21. **print**('[l] load Inventory from file\n[a] Add CD\n[i] Display Current Inventory')
22. **print**('[d] delete CD from Inventory\n[s] Save Inventory to file\n[x] exit')
23. strChoice = input('l, a, i, d, s or x: ').lower()  # convert choice to lower case at time of input
24. **print**()
26. **if** strChoice == 'x':
27. # 5. Exit the program if the user chooses so
28. **break**
29. **if** strChoice == 'l':
30. # TODO Add the functionality of loading existing data
31. **pass**
32. **elif** strChoice == 'a':  # no elif necessary, as this code is only reached if strChoice is not 'exit'
33. # 2. Add data to the table (2d-list) each time the user wants to add data
34. strID = input('Enter an ID: ')
35. strTitle = input('Enter the CD\'s Title: ')
36. strArtist = input('Enter the Artist\'s Name: ')
37. intID = int(strID)
38. lstRow = [intID, strTitle, strArtist]
39. lstTbl.append(lstRow)
40. **elif** strChoice == 'i':
41. # 3. Display the current data to the user each time the user wants to display the data
42. **print**('ID, CD Title, Artist')
43. **for** row **in** lstTbl:
44. **print**(\*row, sep = ', ')
45. **elif** strChoice == 'd':
46. # TODO Add functionality of deleting an entry
47. **pass**
48. **elif** strChoice == 's':
49. # 4. Save the data to a text file CDInventory.txt if the user chooses so
50. objFile = open(strFileName, 'a')
51. **for** row **in** lstTbl:
52. strRow = ''
53. **for** item **in** row:
54. strRow += str(item) + ','
55. strRow = strRow[:-1] + '\n'
56. objFile.write(strRow)
57. objFile.close()
58. **else**:
59. **print**('Please choose either l, a, i, d, s or x!')

Figure 1- Sample Code to edit:

Now, since the first part of the assignment is to replace the data inner structure with dictionaries, the first step is to create a dictionary. Firstly, we must create our dictionary. In the module 5 document, Biesinger explains that the syntax of a dictionary is: Dict\_name ={key:value}. NOTE: It is important to note that dictionaries in Python need to have curly brackets, while arrays do not have curly brackets. Additionally, another factor we must take into consideration if we want to have multiple keys or values in the dictionary, we need to use “,” as a separator between each key:value. With all of this in mind, now we can look at the code that we need to edit.

As we sift through the lines of code, we should look for where in the code there are inner data structures. In analyzing the code, we will find that underneath the elif strChoice A, or line 32 of the code in figure one. In line 38 specifically, we see a row for the 1sttbl, our list of lists defined in the variables. This is where we can replace this row with a dictionary row. Keeping our discussion of the syntax in the paragraph above, here is a starting point for our syntax:

1. Dicrow1 = {‘key’: value, ‘key’: value, ‘key: value’}

For our values, we can use the variables that have been identified in the elif strChoiceA, starting in line 32 of the code in figure one above as values. So, the variables are intID, strTitle, strArtist. So, now that we know what variables we should use, our new code should look like: dicRow1 = {'id': intID, 'title': strTitle, 'artist': strArtist}.

Now that we have replaced our inner data structure by dictionaries, we need to add functionality to append to the 1stbl, defined as our lists of lists in the variables portion of the code in Figure 1. We see in line 39 that is displayed in Figure 1, that there is an append command in place. However, this append command is currently applicable to the 1stRow that was previously in the code, which was structured as a list. Since we have replaced this list with a dictionary row, we must adjust our append command to be applicable to the dictionary row that we just created. So, the only change to the syntax we need is to replace the reference to 1strow to Dicrow1. So, our new append command should look like: lstTbl.append(dicRow1)

So now, our new lines 38 and 39 should look like:

1. dicRow1 = {'id': intID, 'title': strTitle, 'artist': strArtist}
2. lstTbl.append(dicRow1)

**Figure 2: dictionary and append command to append dictionary into the list of lists**

Essentially, what is happening in this line of code within the context of our elif strChoice A, we have changed the code so the user is now inputting their data into a dictionary. From there, we are appending those results in our list of lists, which was defined as 1stTbl in the variables portion of the code.

An important factor, since we have replaced our reference to a list with dicRow1, we need to make sure that if there are any other references to our previous row name, we must adjust it so it references to dicRow1. So, the next line of code that we need edit is in line 44. That line of code currently reads as:

1. **print**(\*row, sep = ', ')

We need to edit this line of code, because the context of the code is saying that it will print the input of each row, separated by a comma. However, since we have converted our row into a dictionary, we need to change this reference to match our adjustment. So, this line should now look like:

1. **for** row **in** lstTbl:
2. **print**(row['id'],row['title'], row['artist'], sep=',')

The reason why it needs to be changed is because this is the proper way to iterate through a line in the dictionary. Essentially, the print function is saying that it will print each id, title and artist input in each dictionary row, separated by a comma.

**Add Functionality of loading existing data:**

Now, the next objective we must complete is we need to give the user the ability to load existing data from a .txt file. While there are several different ways that we can solve this problem, the way that I will presenting is using json.

The first step to utilizing json is we need to import it, as it is a separate library in Python. The way that we import additional libraries in python is with the following syntax: import library\_name. So, in this case, our final line of code is import json. NOTE: we will begin our import in the first line of our code, outside of where we declare our variables and our loops, in line 1 of our script.

The next step is to read and load the existing data from a .txt file. The way that we will be discussing how to do this is through the read .txt file command. The syntax of this command is like writing to a .txt file in function. The first part of the syntax is to use an open function, which goes as follows: with open('strFilename', 'r') as file:

Essentially, what this code is saying that it will open the filename, and r stands for the mode that we want to pass the file through. R stands for read and so that is the mode that will help us complete this task. The next part is to write the command that will allow the user to load the file into our list of lists. The best way to do this is to set a load command within the with loop, that will allow for the user to load the existing data from the .txt file into our list of list. The syntax to approach this task is as follows: lstTbl = json.load(file)

What this line of code is doing is it is using the json.load command, to load the data from the strFilename, which is defined as ‘CDInventory.txt’. Since we set it “as” file, we can use the name file to load the data into our lstTbl. Than, we can use the append command to append the data to our lists of lists.

So, the final code for this command should look like:

1. with open('strFileName', 'r') as file:
2. lstTbl = json.load(file)
3. lstTbl.append(lstTbl)

**Figure 3 – json, loading existing data**

1. Biesinger, Dirk, “FDN\_Py\_Module\_05.pdf”, Foundations of Programming: Python, Module 5, August 10th, 2020
2. <https://stackabuse.com/reading-and-writing-json-to-a-file-in-python/>, August 10th, 202

**Add Functionality of Deleting an entry:**

The next objective of this assignment is to create functionality that will allow the user to delete an entry that they choose. So, the first step that we need to take is to sift through the code to find the option that the user should select that will allow them to delete an entry that they choose. So, it looks like option the user would select in the menu to delete is on line 45 in the first figure. The line currently reads as: **elif** strChoice == 'd'

There are several different ways that we can approach this problem. The way that we will utilize is a for loop, that will iterate through our list of lists. Building off of this, we will add an if loop, which will say that if there is a data point that matches the user input, it will delete the entire row for that entry.

To successfully complete this, the first step we need is to create functionality that will allow the user to input which id and it’s corresponding data they would like to delete. The best way to do this is with the following syntax: delitem = int(input("Please enter in the item you want to delete: "))  If we add this underneath the elif strChoice == ‘d’, once the user enters in ‘d’, it will ask for the user to input the ID number of the row of data the borrower would like to delete.

Before continuing further, I want to note that this step did give me some challenges. Initially, I had created this step by using the clear() command, assuming that this would accomplish the task at hand. However, after further research and trial and error, what I realized through is that while this command will delete entries, it will delete all the entries in the dictionary. So, the step we discussed in the previous paragraph is paramount, as this will require the user to enter in the id they want to delete.

Now, the next step we need to add is a functionality that will iterate through our list of lists. The best way to do this is through a for loop. So, the syntax for our for loop would look like:

**for** item **in** range(len(1stTbl)):

Now, the next step would be to add an if loop, that anytime there is an entry that matches the id the user inputs, it will delete that entry. So, the best way to approach that portion is with this syntax:

**if** lstTbl[i]['id'] == delID:

Essentially, what this line of code is saying, that anytime there is an ID entry that matches the input the user put in, it will delete all the data that is associated with that ID.

The next line of code that we need to add is the command that will carry out the delete command. That line of code should look like: del lstTbl[i]

So, now that we have broken out the lines of code for this command, the final code should look like:

1. delID = int(input("Please enter in the ID you want to delete: "))
2. **for** i **in** range(len(lstTbl)):
3. **if** lstTbl[i]['id'] == delID:
4. **del** lstTbl[i]
5. **break**:

**Figure 4: Delete entries by individual ID command**

1. <https://www.geeksforgeeks.org/python-removing-dictionary-from-list-of-dictionaries/>, August 14th, 202

**Adding Functionality to be able to save to a .txt file if the User chooses to do so:**

Now, the final part of our assignment is to add functionality allowing the User to save to a .txt file, if they choose to do so. Looking at the existing code, on line 49 there is pseudocode that has instructions to add this function, so this is where we will begin.

The first step is to write a script that will open the .txt file or create one if the file is not available. This script should also than allow the user to write into the file. Then, it should allow the user to save data to the .txt file. The way that I will present on how to solve this problem is with the following syntax:

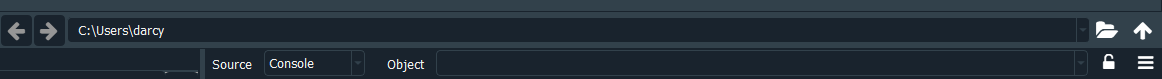
1. with open('strFilename', 'w') as file:
2. file.write(json.dumps(lstTbl))

**Figure 5 – Saving to a .txt file**

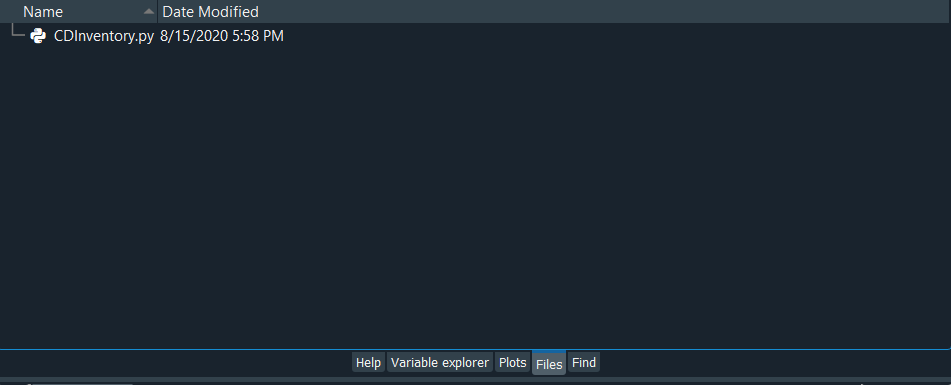
Essentially, what this code is doing is it is opening our strFilename, which we have defined as ‘CDInventory.txt’. The w mode is the mode that we want to use when we are opening this file. The following code in line two is the actual command that allows us to write into the file while utilizing json, which is the library that we have imported in our previous step.

**Verifying the code works in our IDE:**

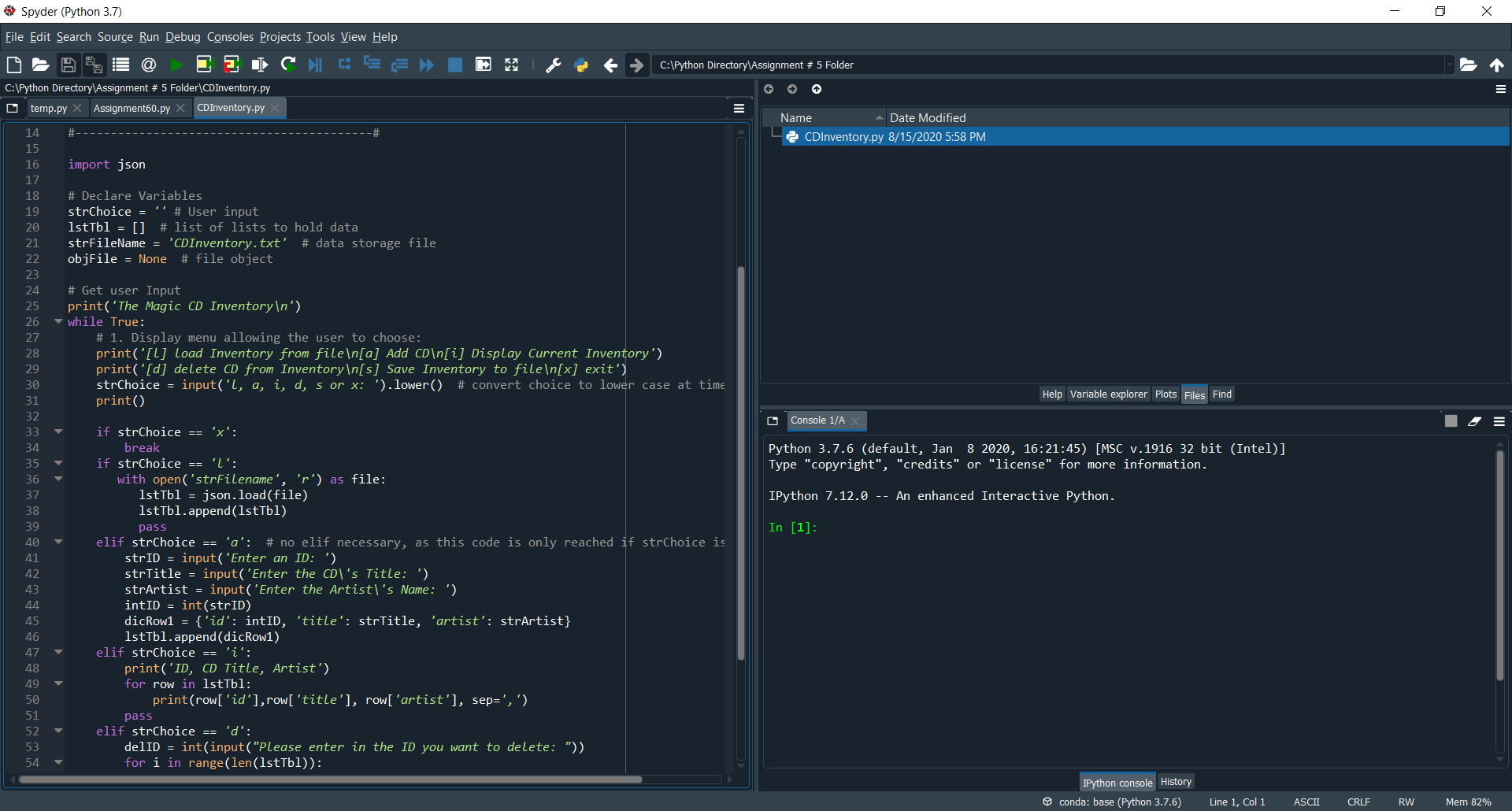
An important part of programming is to ensure that our code is working, and so we will now discuss how to verify in our IDE that the program is working. The first step we need to take is to open our IDE, which we have been using Spyder. Once we have opened our IDE and we are at our home screen, the next step is to change our directory. On the right-hand side of our home screen, the next step is to click on the folder button that you can see in the figure below.



**Figure 6 - Changing directories in your IDE**

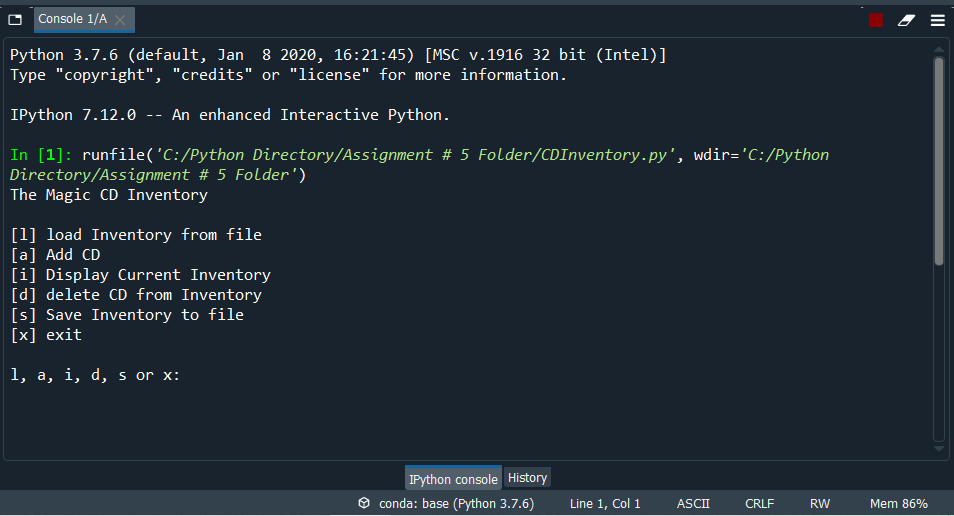
Once you click on the folder, than you can directly select where your .py file is located. Than you will need to click on the Files button as seen in the figure below. It will than pull up the files that are housed in that folder. From there, you can double-click on your .py file and that should automatically post your .py file into your IDE. 

**Figure 7 – locating your .py file**



**Figure 8 – verify your script is ready to run in your IDE**

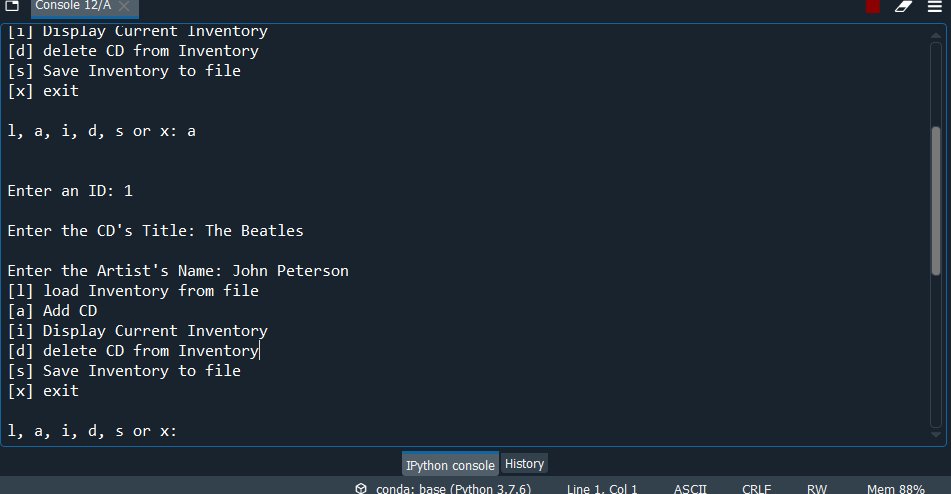
Now, to run your script you can click on the green play button in the toolbar of your IDE in the figure above. Once you run your code, you should see a menu appear in your console:



**Figure 9 – Menu of CDInventory.py**

Now, let’s test each of these options to ensure the code is working properly.

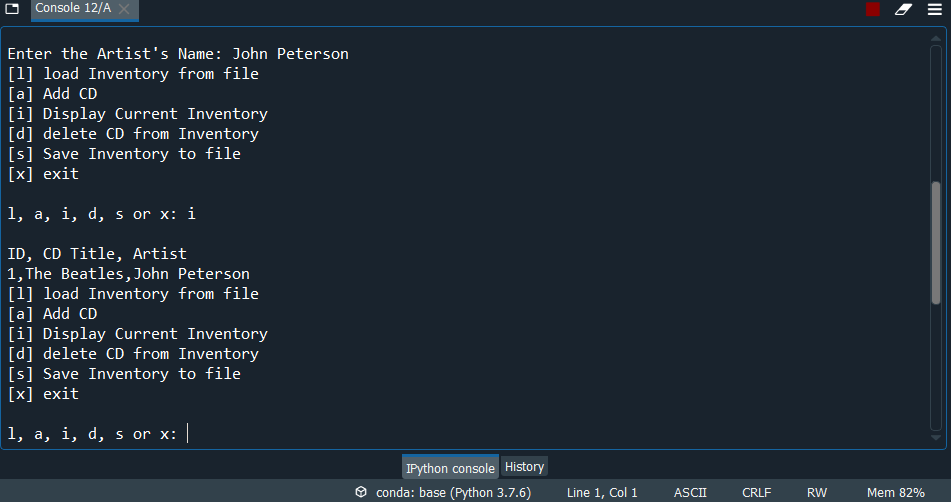
The first option that we are testing is adding a CD to our inventory, below is a confirmation that it is working:



**Figure 10 – Add CD option**

Remember, our ID input is an int, so that is why we have entered in a number in our example. The second part of this elif option is to ask for the user to enter in a CD title, which we have entered in successfully. The third and final input is to enter in the artist’s name, which we have been able to successfully enter.

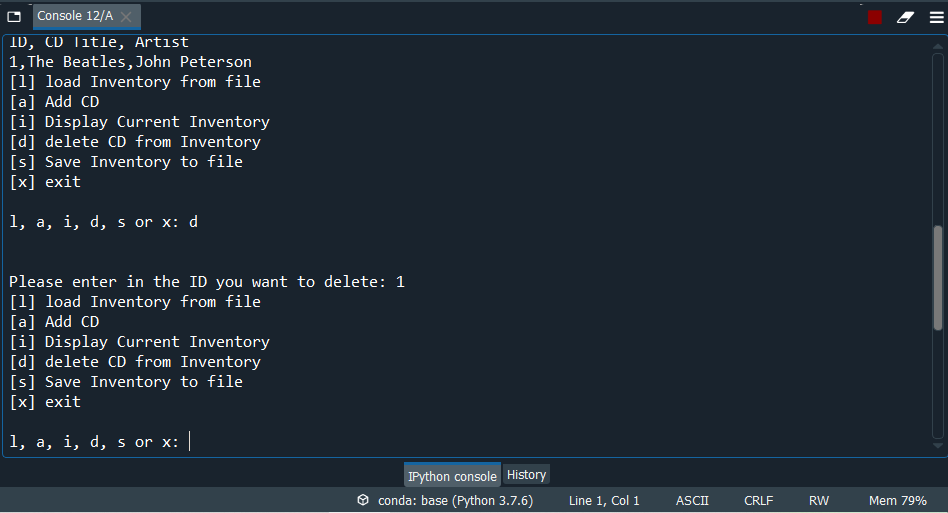
The second input option that we are testing is displaying our current inventory. As we entered in our input, below is the results:



**Figure 11 – Display Current Inventory**

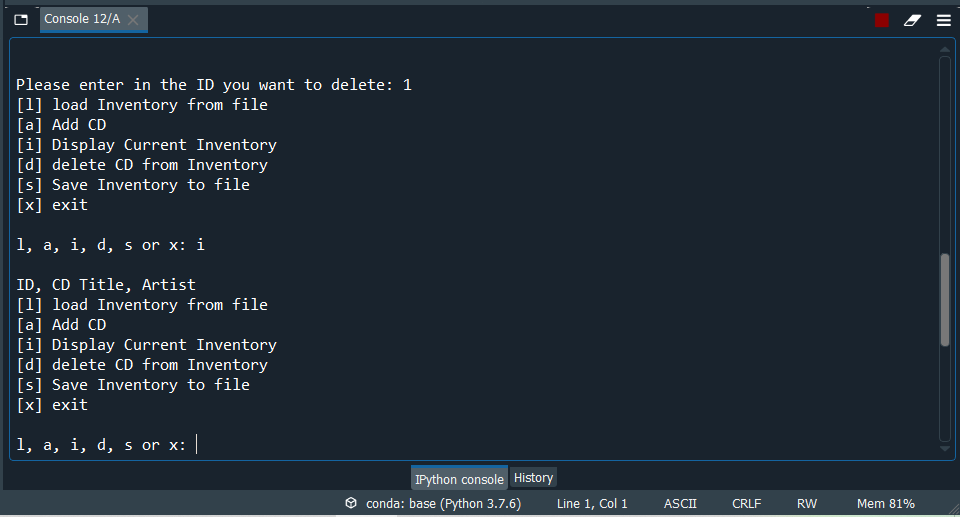
We can see that the inventory has displayed based off our previous entry. The inputs that we have added in our script is currently being displayed.

The next option that we are going to test is the user’s ability to delete a CD from the inventory.



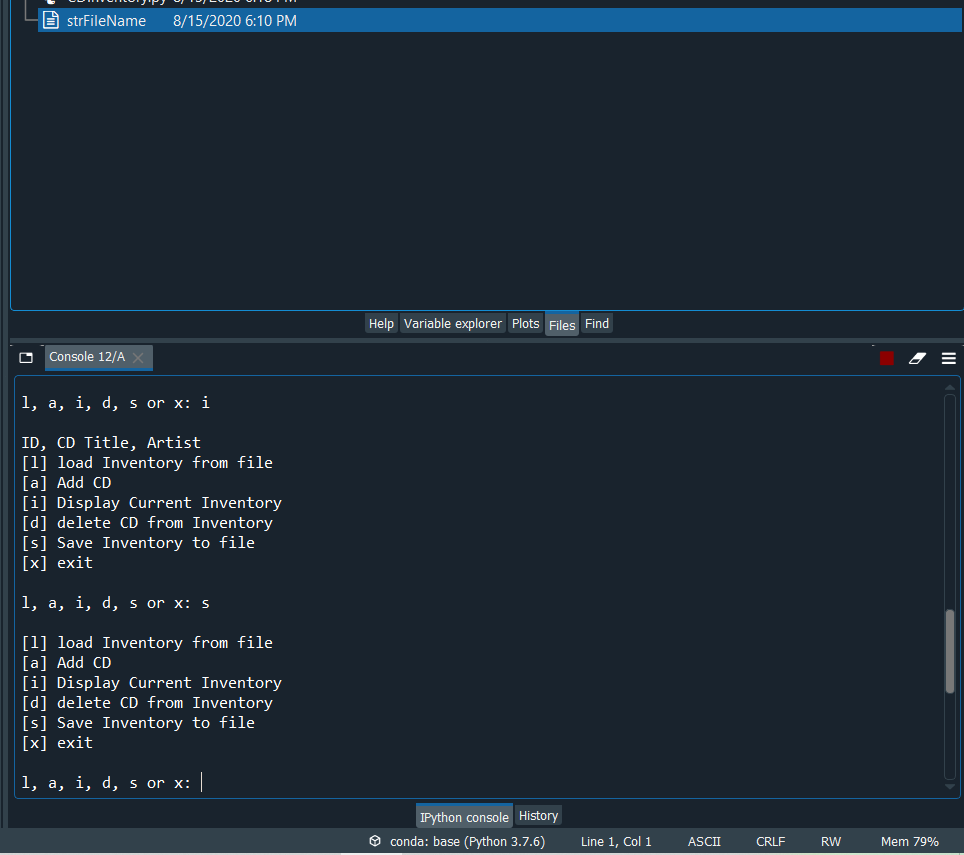
**Figure 12 – Delete CD from Inventory**

Now, we will display our inventory again to confirm that the file had been successfully deleted.



We can see that our previous entry is no longer in our inventory. Success!

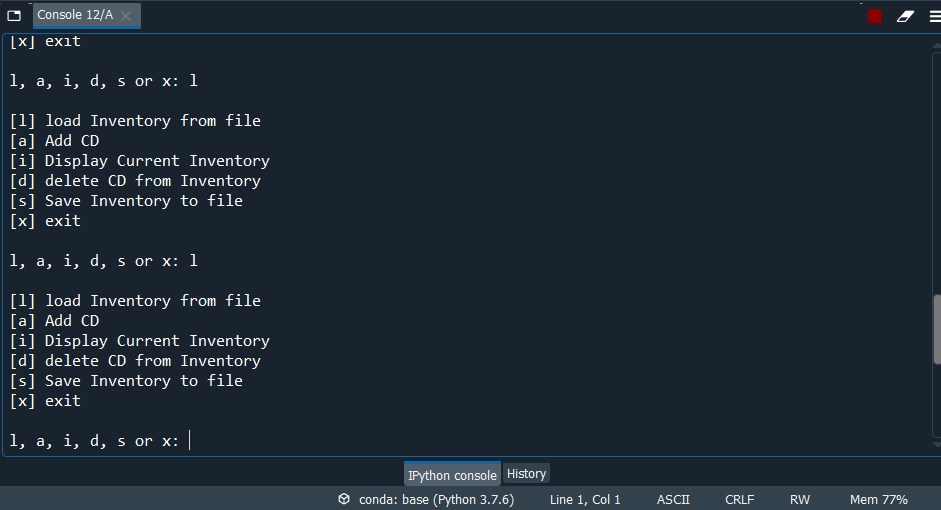
Now, the next option we need to test is the s option, for the user.



**Figure 13 – Save Inventory to File**

We can see that once we select option s, there is a new .txt file ‘strFileName’ created in our directory. Since there was not an existing .txt file previously, the command that we have for this elif option will create the file if there isn’t one already in place.

Now, we need to test option l. Below is the results, which we have ran into no syntax errors. NOTE: it is important to note that if the user wants to utilize this option, we need to save our inventory data to a .txt file first, as this command is built to read the data from the .txt file created in our [s] option.



**Figure 14 – Load Inventory to File**

GitHub:

Now, in this portion I will introduce Github. Github allows for programmers to share different versions of their code. It’s a great way to collaborate with other programmers, and another resource to gain insights and clarifications on how to overcome hurdles that come up when programming.

The first step to start using GitHub is you will need to create an account. Fortunately, to use GitHub is completely free and is relatively easy to sign up for. The first step is to follow this link here: [www.github.com](http://www.github.com)

Once you are at this website, you will be prompted to sign up for an account. Once you have completed the sign up, you will receive an verification email request from GitHub to verify the email you submitted.

Once you have completed your account, you can now start sharing your files through what is called a repository. Once you have created your repository, you can than start to upload files to the repository. Once you have completed the upload of your files and scripts, you can than share your repository link with your friends, fellow programmers or in your professional portfolios and profiles! For our assignments purpose, I will share the files to the canvas discussion board, by copying and pasting the repository link into the Canvas discussion board.

4.) Biesinger, Dirk, “FDN\_Py\_Module\_05.pdf”, Foundations of Programming: Python, Module 5, August 10th, 2020

Summary:

In conclusion, now I have gained a deeper understanding of how to work with existing code, specifically how to replace inner data structures with dictionaries. Also, I have gained a deeper understanding of the syntax of dictionaries and how dictionaries operate in comparison to list. Furthermore, I also learned more about GitHub and how to successfully use and upload submissions to GitHub. Also, I have gained a basic understanding of how the json library operates. One of the hurdles that I faced was being able to create a functionality that would allow a user to be able to delete individual entries. Initially, I had difficulties with this command as I could only create a command that would clear all the entries. However, after further research I was able to find the answer that would allow the user to pick the individual entry that they want to delete, while keeping the keys in the dictionary. With every assignment, module and class, I have grown to develop a deeper appreciation and understanding of Python programming.

Appendix:

Biesinger, Dirk, “FDN\_Py\_Module\_05.pdf”, Foundations of Programming: Python, Module 5, August 10th, 2020

<https://www.geeksforgeeks.org/python-removing-dictionary-from-list-of-dictionaries/>

[www.planetb.ca](http://www.planetb.ca), “Syntax Highlighter”

<https://stackabuse.com/reading-and-writing-json-to-a-file-in-python/>